**Expr 8: Semaphore**

**Semaphore code:**

#include <stdio.h>

#include <stdlib.h>

#include <pthread.h>

#include <semaphore.h>

#include <unistd.h>

#define BUFFER\_SIZE 5

#define PRODUCE\_COUNT 10

int buffer[BUFFER\_SIZE];

int in = 0, out = 0;

sem\_t empty, full, mutex;

void \*producer(void \*arg) {

    for (int i = 0; i < PRODUCE\_COUNT; i++) {

        int item = i + 1;

        sem\_wait(&empty);

        sem\_wait(&mutex);

        buffer[in] = item;

        printf("Producer produced: %d\n", item);

        in = (in + 1) % BUFFER\_SIZE;

        sem\_post(&mutex);

        sem\_post(&full);

        sleep(1);

    }

    pthread\_exit(NULL);

}

void \*consumer(void \*arg) {

    for (int i = 0; i < PRODUCE\_COUNT; i++) {

        sem\_wait(&full);

        sem\_wait(&mutex);

        int item = buffer[out];

        printf("Consumer consumed: %d\n", item);

        out = (out + 1) % BUFFER\_SIZE;

        sem\_post(&mutex);

        sem\_post(&empty);

        sleep(1);

    }

    pthread\_exit(NULL);

}

int main() {

    pthread\_t prodThread, consThread;

    sem\_init(&empty, 0, BUFFER\_SIZE);

    sem\_init(&full, 0, 0);

    sem\_init(&mutex, 0, 1);

    pthread\_create(&prodThread, NULL, producer, NULL);

    pthread\_create(&consThread, NULL, consumer, NULL);

    pthread\_join(prodThread, NULL);

    pthread\_join(consThread, NULL);

    sem\_destroy(&empty);

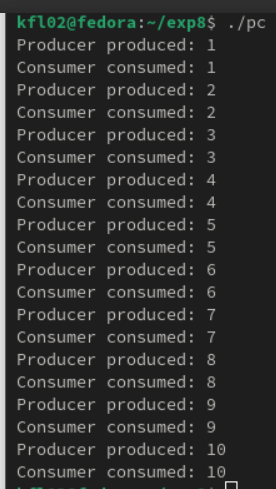
    sem\_destroy(&full);

    sem\_destroy(&mutex);

    return 0;

}

**Output:**

****

**Result:**

Thus the Semaphore Code is implemented in fedora using the C language